

[INICIAR SESIÓN](#)[NUESTROS PLANES](#)[TODOS LOS CURSOS](#)[FORMACIONES](#)[CURSOS](#)[PARA EMPRESAS](#)[ARTÍCULOS DE TECNOLOGÍA > DEVOPS](#)

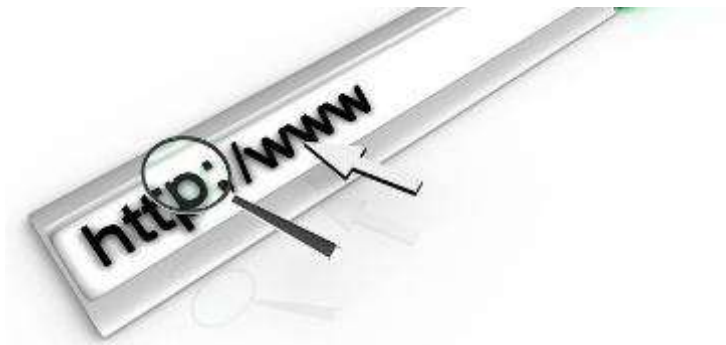
¿Cual es la diferencia entre HTTP y HTTPS?



rafael-nercessian

09/09/2021

Es posible que hayas visto sitios web que tienen un símbolo de candado en su URL y el nombre HTTPS al lado.



Para entender esta diferencia, consideremos la siguiente aplicación web en la que un usuario de una empresa (llamémoslo de Juan) está usando solo el protocolo HTTP, es decir, el modo "No seguro":



Tenga en cuenta que ingresa su correo electrónico y contraseña, luego hace clic en el botón Iniciar sesión (Login). Hasta ahora no hemos visto ningún problema, ¿verdad?


Conociendo el escenario problemático

Sin embargo, esta aplicación se encuentra dentro de un entorno corporativo, por lo que hay varios empleados. Entonces uno de estos otros empleados, lo llamaremos Pedro, decide hacer algunas pruebas en la red corporativa.

En estas pruebas, Pedro cambia los protocolos de comunicación entre Juan que está accediendo a la aplicación web y el enrutador, y luego Pedro termina en medio de esta comunicación entre Juan y el enrutador.

Entonces Pedro, por curiosidad, acaba instalando en su máquina un programa capaz de analizar los protocolos que transitan en la comunicación entre Juan y el enrutador. Luego, configura el programa para analizar todas las comunicaciones a través del protocolo HTTP.

En ese mismo momento, cuando Juan intenta acceder nuevamente al sistema web, aparece el siguiente resultado para Pedro, nuestro curioso empleado:



```
Stream Content:
POST /login HTTP/1.1
Host: ejemplo-login-fraco-alura.herokuapp.com
Connection: keep-alive
Content-Length: 54
Cache-Control: max-age=0
Origin: http://ejemplo-login-fraco-alura.herokuapp.com
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/537.36
Content-Type: application/x-www-form-urlencoded
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Referer: http://ejemplo-login-fraco-alura.herokuapp.com/login
Accept-Encoding: gzip, deflate
Accept-Language: pt-br;q=0.8,en-us;q=0.6,en;q=0.4
Cookie: 1S655D0WID=704A76420894464C880E8C0924DF5C8C

401 Unauthorized
Content-Type: text/html; charset=UTF-8
Content-Length: 90
Date: Sun, 26 Feb 2017 14:14:17 GMT
Via: 1.1 vegur

GET /login HTTP/1.1
Host: ejemplo-login-fraco-alura.herokuapp.com
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Referer: http://ejemplo-login-fraco-alura.herokuapp.com/login
Accept-Encoding: gzip, deflate, sdch
Accept-Language: pt-br;q=0.8,en-us;q=0.6,en;q=0.4
Cookie: 1S655D0WID=704A76420894464C880E8C0924DF5C8C

HTTP/1.1 200 OK
Connection: keep-alive
Server: Apache-Coyote/1.1
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Type: text/html; charset=UTF-8
Content-Language: pt-br
Content-Length: 402
Date: Sun, 26 Feb 2017 14:14:17 GMT
Via: 1.1 vegur

<!DOCTYPE html>
<html>
<head>
```

Pero, ¿cómo fue posible obtener estas informaciones?

Entendiendo el principio del protocolo HTTP

Las comunicaciones a través de una red se realizan mediante un conjunto de reglas y procedimientos llamados protocolo [Enlace](#). El protocolo HTTP, Hypertext Transfer Protocol, se desarrolló inicialmente para permitir la comunicación con un servidor web, al principio, el método existente era simplemente "tomar" (**GET**) el contenido de un servidor.

Con el tiempo, surgió la necesidad no solo de solicitar informaciones al servidor, sino también de transmitir las informaciones. Actualmente sabemos que en la Web no solo pedimos, sino que también enviamos informaciones, ¿verdad? Como es el caso de una compra por Internet. Pero, ¿qué sucede si se trata de un dato sensible como los datos y contraseña de la tarjeta de crédito?

Al analizar el nombre del protocolo HTTP, vemos que es un protocolo para transferir texto, ¡por lo que los datos y la contraseña de la tarjeta de crédito también se pasan como texto al servidor!

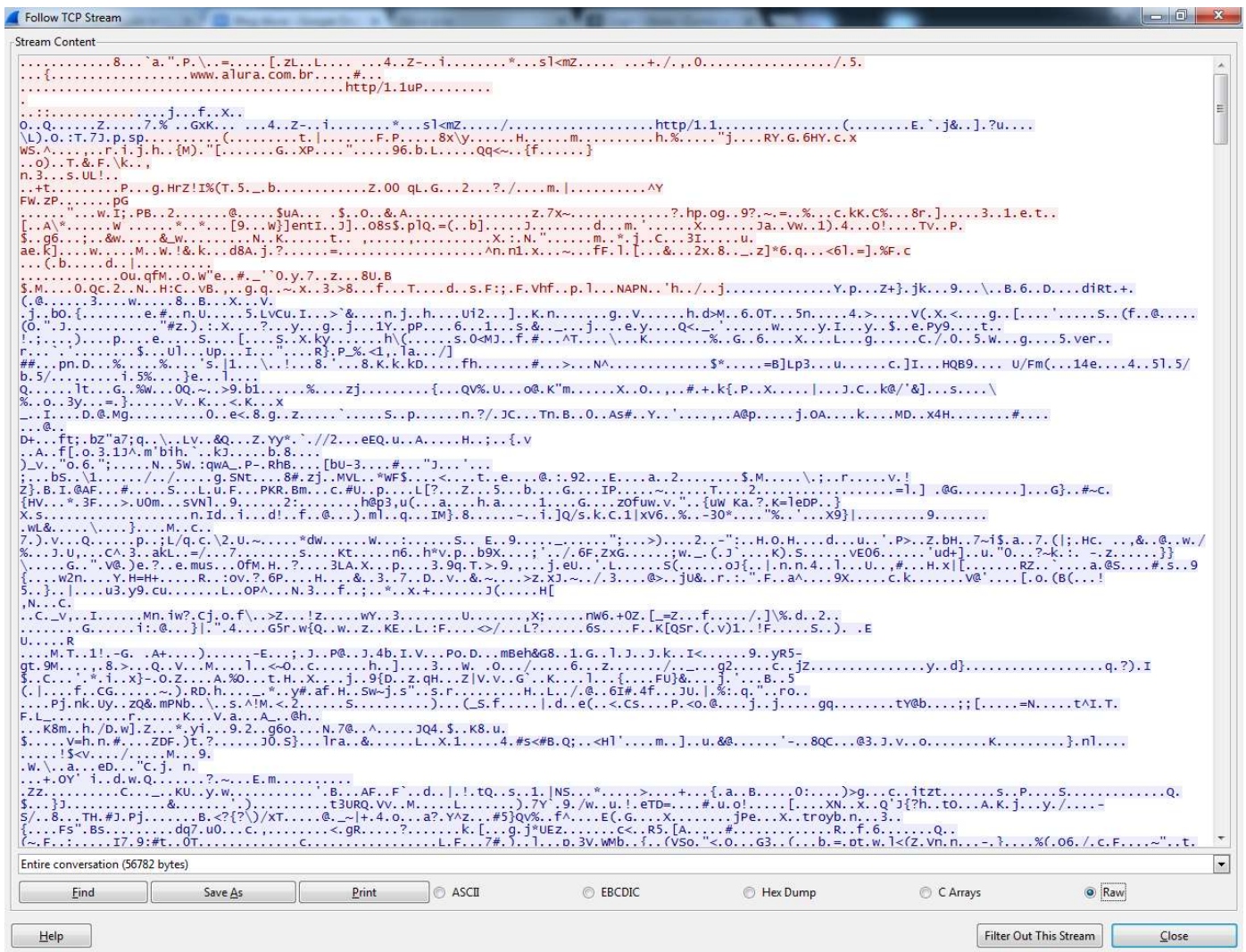
Esto significa que si alguien analiza estas informaciones, podrá ver cuáles son los datos y la contraseña de la tarjeta de crédito. Por lo tanto, cualesquier aplicaciones web que utilicen el protocolo HTTP y necesiten pasar datos sensibles tienen una vulnerabilidad para el cliente final.

Siendo consciente de todo este problema, todo sitio web que opera con el protocolo HTTP necesita, de alguna manera, encontrar alguna forma de protegerse, ¿verdad? ¡Ahí es cuando HTTPS entra como un gran aliado! Pero, ¿qué hace el protocolo HTTPS para solucionar esta vulnerabilidad?

Conociendo el protocolo HTTPS

Ahora, en lugar de analizar un sitio web "No seguro", es decir, con el protocolo HTTP, haremos el análisis en un sitio web "Seguro" que funcione con HTTPS.

En nuestro ejemplo, consideraremos la página de inicio de sesión de Alura usando los mismos datos que usamos en la aplicación que usaba HTTP. Veamos el resultado:



Date cuenta de que la información no tiene ningún sentido para nosotros los humanos.

¿Qué pasó esta vez?

HTTPS hizo una transformación en estos datos enviados por el cliente y, por lo tanto, no fue posible descubrir los datos enviados por la víctima. Esta transformación de datos que resulta incomprensible para los humanos la caracterizamos como un proceso criptográfico.

Ahora bien, ¿cómo se realiza este proceso de cifrado HTTPS?

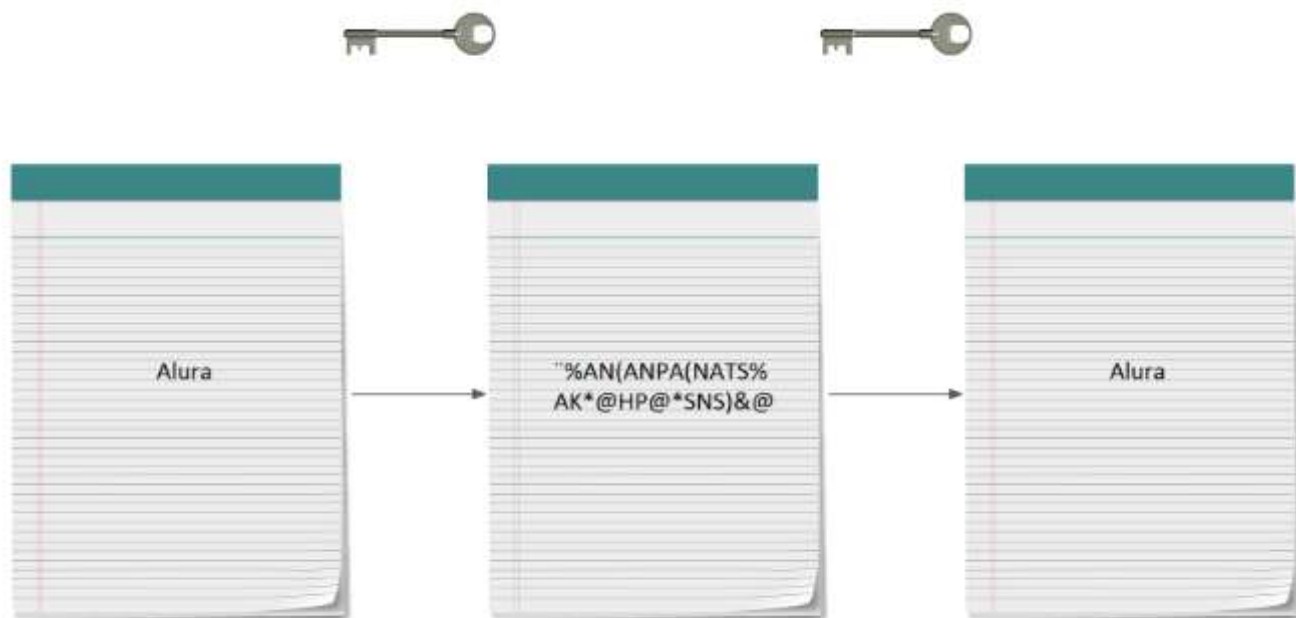
Comprendiendo el proceso HTTPS

Analicemos el siguiente caso, imaginemos que dos personas, Roberto y Luísa quieren enviar una carta, Roberto inicia el proceso de comunicación enviando esta carta a Luísa, pero esta carta es confidencial y no debe ser leída por nadie más que Luisa.

Proceso de claves simétricas

Para evitar que nadie más lea el contenido de esta carta, Roberto decide ponerla en una caja con candado, la cierra con llave y luego se dirige a la oficina de correos para enviar la carta.

Luísa recibe la caja y usa una copia de la llave idéntica a la de Roberto para abrir la caja y leer el contenido de la carta. Para responder al mensaje recibido, Luísa vuelve a poner el candado y utiliza la misma llave para cerrar la caja y enviarla a través de la oficina de correos.



Tenga en cuenta que solo tenemos una clave en este proceso, la misma clave se usa para bloquear la caja (cifrar) y desbloquear la caja (descifrar). Cuando tenemos la misma clave que se utiliza para cifrar y descifrar datos, llamamos a este proceso de cifrado de **clave simétrica**. Ejemplos de algunos algoritmos que utilizan este sistema de clave simétrica:

- **DES (Data Encryption Standard)**
- **AES (Advanced Encryption Standard)**

Dado que la misma clave se utiliza para realizar el cifrado y el descifrado, es necesario encontrar una manera de enviar esta clave al destinatario (si aún no la tiene) para que pueda desbloquear la caja y leer el contenido de la carta (desencriptar).

Pero mira lo que tenemos, si se usa la misma clave para bloquear (encriptar) y desbloquear (desencriptar), ¿qué pasaría si alguien lograra interceptar esa clave? Esa persona podría usarla para poder ver información pasada y seguiríamos teniendo el mismo problema, ¿verdad? ¿Y ahora?

Por lo tanto, solo el uso de este proceso de claves simétricas durante los pasos de comunicación puede no solucionar completamente el problema y, por lo tanto, también se desarrolló una nueva forma de criptografía para complementar el uso de claves simétricas. Volvamos al ejemplo anterior.

Proceso de claves asimétricas

Tenemos a Roberto y Luísa, Roberto quiere enviarle una carta con contenido confidencial a Luísa. La idea ahora es que Roberto use un candado especial y ese candado tendrá 3 posiciones:

1 - bloqueado 2 - desbloqueado 3 - bloqueado



Además, Roberto ahora tendrá dos claves, la primera clave solo se puede usar en orden ascendente (posición 1, 2 y 3) y la otra clave solo se puede usar en posición descendente (posición 3, 2 y 1).

Roberto decide cerrar la caja con la primera clave, la que logra accionar el candado en orden ascendente. De esta forma, para bloquear la caja, ahora estará en la posición 3. Esta clave que Roberto no enviará a nadie, es su clave única, llamaremos a esta clave de **clave privada**.

La segunda clave que opera en orden descendente (posición 3, 2 y 1), se utilizará para enviarla a Luísa, ya que estamos enviando esta segunda clave a otra persona, la llamaremos **clave pública**.

Cuando la caja llegue a Luísa, se bloqueará en la posición 3. Con la clave pública que recibió Luísa, puede operar en orden descendente y devolver el candado a la posición

2 (posición desbloqueada) y abrir la caja, leyendo así el contenido de la carta.

Luísa, al responder al mensaje, lo bloqueará con la clave pública. Recuerda que la clave pública solo funciona en sentido descendente, por lo que al estar en la posición 2 (desbloqueada), solo puede ir a la posición 1 para volver a bloquearla. Una vez que Roberto recibe la caja, puede utilizar su clave privada (la que se guardó para sí mismo y no la distribuye a nadie y solo funciona en orden ascendente) para devolver la cerradura a la posición 2 (desbloqueada) y así poder abrir la contenido del mensaje.



Tenga en cuenta que ahora tenemos dos claves, una clave privada y una clave pública, donde cuando una clave se usa para bloquear la caja (cifrar), la otra clave se usa para desbloquear la caja (descifrar). Cuando tenemos dos claves, llamamos a este proceso de encriptación de **clave asimétrica**. Ejemplos de algunos algoritmos que utilizan este sistema de claves asimétricas:

- RSA (Siendo las iniciales de los apellidos de sus desarrolladores: Ron Rivest, Adi Shamir y Leonard Adleman)
- DSA (Digital Signature Algorithm)

Implementación del protocolo de seguridad HTTPS

HTTPS utilizará los principios de claves simétricas y asimétricas durante el proceso de comunicación. Dentro de HTTPS existe un protocolo encargado de insertar esta capa de cifrado, conozcamos algunos de ellos:

SSL

El primer protocolo desarrollado para este propósito fue creado por la empresa Netscape y se denominó **Secure Sockets Layer 1.0**, o más conocido como SSL v1.0. El protocolo **SSL** sufrió modificaciones y mejoras hasta llegar a su última versión, SSL v3.0.

Sin embargo, a lo largo de los años, se descubrieron algunas vulnerabilidades en este protocolo, una de estas vulnerabilidades podría exponer informaciones encriptada y terminó llamándose Poodle (Padding Oracle on Downgraded Legacy Encryption) para más detalles sigue el enlace del gobierno norteamericano alertando sobre eso [vulnerabilidad descubierta en el protocolo SSL](#).

TLS

Dado que el protocolo SSL v3.0 presentó estas vulnerabilidades, fue necesario desarrollar un nuevo tipo de protocolo para asignar esta capa de cifrado de una manera más segura.

Este nuevo protocolo se denominó Transport Layer Security, TLS 1.0 sufriendo adaptaciones y mejoras a lo largo del tiempo, llegando a la versión más actual hasta el momento de escribir esta publicación, la versión TLS 1.2.

El principio de uso de TLS es unir las fortalezas de las claves asimétricas y simétricas durante el proceso de comunicación. Dado que las claves asimétricas tienen un proceso de cifrado que involucra dos claves distintas, por lo tanto, hay un tiempo de procesamiento más largo. Es por eso que TLS usa este proceso en la fase de autenticación entre el cliente y el servidor.

Luego, hace uso de claves simétricas que tienen un proceso de encriptación con una sola clave, por lo que requiere menos tiempo para su procesamiento.

Dado que es posible que tengamos una gran cantidad de datos transmitidos, las claves simétricas serán más rápidas para procesar toda esta información y se utilizarán en la fase de transporte de datos, también conocida como **bulk**.

Puedes leer también:

- [HTTP: Diferencias entre GET y POST](#)
- [HTTP: Desmitificando el protocolo Web](#)

ARTÍCULOS DE TECNOLOGÍA > DEVOPS

En Alura encontrarás variados cursos sobre DevOps. ¡Comienza ahora!

SEMESTRAL

US\$49,90

un solo pago de US\$49,90

- ✓ 218 cursos
- ✓ Videos y actividades 100% en Español
- ✓ Certificado de participación
- ✓ Estudia las 24 horas, los 7 días de la semana
- ✓ Foro y comunidad exclusiva para resolver tus dudas
- ✓ Acceso a todo el contenido de la plataforma por 6 meses

¡QUIERO EMPEZAR A ESTUDIAR!

[Paga en moneda local en los siguientes países](#)

ANUAL

US\$79,90

un solo pago de US\$79,90

- ✓ 218 cursos
- ✓ Videos y actividades 100% en Español
- ✓ Certificado de participación
- ✓ Estudia las 24 horas, los 7 días de la semana
- ✓ Foro y comunidad exclusiva para resolver tus dudas
- ✓ Acceso a todo el contenido de la plataforma por 12 meses

¡QUIERO EMPEZAR A ESTUDIAR!

[Paga en moneda local en los siguientes países](#)

Acceso a todos
los cursos

Estudia las 24 horas,
dónde y cuándo quieras

Nuevos cursos
cada semana

NAVEGACIÓN

PLANES

INSTRUCTORES

BLOG

POLÍTICA DE PRIVACIDAD

TÉRMINOS DE USO

SOBRE NOSOTROS

PREGUNTAS FRECUENTES

¡CONTÁCTANOS!

¡QUIERO ENTRAR EN CONTACTO!

BLOG

PROGRAMACIÓN

FRONT END

DATA SCIENCE

INNOVACIÓN Y GESTIÓN

DEVOPS

AOVS Sistemas de Informática S.A
CNPJ 05.555.382/0001-33

SÍGUENOS EN NUESTRAS REDES SOCIALES



ALIADOS



En Alura somos unas de las Scale-Ups seleccionadas por Endeavor, programa de aceleración de las empresas que más crecen en el país.



Fuimos unas de las 7 startups seleccionadas por Google For Startups en participar del programa Growth Academy en 2021

POWERED BY

CURSOS

Cursos de Programación

Lógica de Programación | Java

Cursos de Front End

HTML y CSS | JavaScript | React

Cursos de Data Science

Data Science | Machine Learning | Excel | Base de Datos | Data Visualization | Estadística

Cursos de DevOps

Docker | Linux

Cursos de Innovación y Gestión

Productividad y Calidad de Vida | Transformación Ágil | Marketing Analytics |
Liderazgo y Gestión de Equipos | Startups y Emprendimiento